# Formulario HTML

- **<form>**: Contenedor que agrupa los controles del formulario.
- **<input>**: Campo de entrada de texto, checkbox, radio, etc. **<input type="text" name="nombre">**
- **<textarea>**: Campo para entrada de texto en múltiples líneas. **<textarea name="comentarios"></textarea>**
- **<select>**: Menú desplegable.
    - **<options>**: cada una de las opciones del menú.
- **<button>**: Botón interactivo.

Las etiquetas **<fieldset>** y **<legend>** se utilizan para agrupar elementos de formulario relacionados y proporcionar un título o leyenda descriptiva para el grupo.

La etiqueta **<fieldset>** actúa como un contenedor que agrupa campos del formulario

```
<form action="https://www.w3schools.com/action_page.php"
method="post">
```

**Metodo get por defecto, se ve en la busqueda, post no**

**Borde con <fieldset>**

```
<legend>Datos personales</legend>
```

**Multiple ->** seleccion multiple

## Ejemplos:

```
<legend>Datos personales</legend>
        <label for="nombre">Nombre y apellidos: </label>
```

Nombre y apellidos: Juan García Pérez

```
<label for="socio">Socio:</label>
        <select name="socio" id="socio">
<!--Etiqueta option-->
            <option value="value1">Sí<option>
            <option value="value2" selected>No<option>
```

**Selected ->** valor predeterminado

Socio: No

```
<label>Edad: </label>
        <input name="edad" type="checkbox" value="20-39">
20-39
        <input name="edad" type="checkbox" value="40-59">
40-59
        <input name="edad" type="checkbox" value="60-79">
60-79
```

Edad: ☐ 20-39 ☑ 40-59 ☐ 60-79

```
<label for="email">Email: </label>
        <input id="email" name="email" type="email">
```

Email: juan.garcia@perez.com

```
<label for="estudios">Estudios: </label>
        <select name="estudios" id="estudios">
            <optgroup label="Estudios Universitarios">
                <option>Doctorado</option>
                <option>Máster</option>
                <option>Grado</option>
            </optgroup>
            <optgroup label="Ciclo Formativo">
                <option>Grado Superior</option>
                <option>Grado Medio</option>
            </optgroup>
        </select>
```

Estudios: Grado

```
<label for="navegador">Elige tu navegador favorito:
</label>
        <input name="navegador" id="navegador"
list="browsers">
        <datalist name="browsers" id="browsers">
            <option value="Chrome">
            <option value="Firefox">
            <option value="Internet Explorer">
            <option value="Opera">
            <option value="Safari">
            <option value="Microsoft Edge">
        </datalist>
```

Elige tu navegador favorito: Firefox

```
<label for="mensaje">Escribe tu mensaje: </label>
        <textarea rows="10" cols="50"
placeholder="Mensaje" id="mensaje" name="mensaje"
></textarea>
```

Escribe tu mensaje: Hola Mundo.
Esto es una prueba.
Formulario

```
</fieldset>
    <input type="submit" name="enviar" value="Enviar">
</form>
```

Enviar

## Atributos

El atributo **placeholder** muestra un texto de sugerencia en el campo de entrada cuando está vacío. Este texto desaparece cuando el usuario comienza a escribir en el campo.

```
<input type="text" name="correo" placeholder="Ingrese su correo
electrónico">
```

El atributo **required** indica que un campo de entrada es obligatorio y debe completarse antes de enviar el formulario.

```
<input type="text" name="nombre" placeholder="Ingrese su nombre" required>
```

El atributo **disabled** desactiva un campo de entrada, lo que significa que los usuarios no pueden interactuar con él ni modificar su valor. Los campos desactivados no se envían con el formulario.

```
<input type="text" name="nombre" value="John Doe" disabled>
```

El atributo **readonly** hace que un campo de entrada sea de solo lectura. A diferencia de los campos desactivados, los campos de solo lectura sí se envían con el formulario, pero los usuarios no pueden modificar su valor.

```
<input type="text" name="id_usuario" value="12345" readonly>
```

# Video y audio

Existen etiquetas que incorporan (embed) elementos externos a en una página web. Estos elementos pueden ser imágenes, vídeos, audios, mapas, fórmulas, etc...

- **img**. Representa una imagen.
- **iframe**. Representa un contexto anidado de navegación, es decir, un documento HTML embebido.
- **embed**. Representa un punto de integración para una aplicación o contenido interactivo externo que por lo general no es HTML.
- **object**. Representa un recurso externo, que será tratado como una imagen, un sub-documento HTML o un recurso externo a ser procesado por un plugin.
- **video**. Representa un vídeo.
- **audio**. Representa un sonido.
- **source**. Permite a autores especificar recursos multimedia alternativos para los elementos multimedia como <video> o <audio>. Por ejemplo, audios en diferentes idiomas.
- **canvas**, Representa un área de mapa de bits en el que se pueden utilizar scripts para renderizar gráficos.
- **svg**. Define una imagen vectorial embebida.

```
<audio autoplay controls loop muted preload="valor" src="valor">
  Texto para navegadores que no soportan audio…
</audio>
```

Entre los atributos de la etiqueta encontramos los siguientes:

| Atributo | Valor | Descripción |
|---|---|---|
| autoplay | autoplay | El audio comenzará a reproducirse tras su carga. |
| controls | controls | Especifica que se mostrarán los controles (play, stop, ...) |
| loop | loop | El audio se repetirá indefinidamente una vez acabe. |
| muted | muted | El audio estará en silencio. |
| preload | auto<br>metadata<br>none | Especifica el comportamiento del audio una vez se cargue la página. |
| src | *URL* | Origen del audio |

# Color

- Nombre del color: blue, red, orange, green...
- Proporción de cada valor RGB, en código hexadecimal, precedido de almohadilla: #FF0000.
- Proporción de cada valor RGB en decimales, de 0 a 255: rgb(255, 0, 0).
- Proporción de cada valor RGB en porcentajes: rgb(100%, 0%, 0%).
- Proporción de cada valor RGB, más un valor de 0 a 1 (alpha), que indica la transparencia de dicho color: rgba(255, 0, 0, 0.5).

# Texto

**Text-align ->** alinear texto, también justificar (justify)

**Text-decoration ->** subrayado / tachado :

- underline: subrayado típico
- none: quitar subrayado
- overline: subrayado por encima
- line-through: tachado

**Text-transform ->** mayúsculas / minúsculas

- uppercase: mayúsculas
- lowercase: minúsculas
- capitalize: primera letra de cada palabra en mayúscula

**Font-size ->** tamano del texto

- px, pt, %, em (similar al porcentaje, pero tomando 1 como base: 120% = 1.2em)
- x-small, small, medium, large o x-large

**Font-family ->** tipo de fuente: Helvetica, Verdana, "Times New Roman", serif

Como tiene espacios se pone entre " "

**Font-style ->** "cursiva":

- normal: elimina la propiedad

- italic: cursiva
- oblique: oblicua


**Text-weight -> "**negrita", aunque existen múltiples valores, muy pocas fuentes soportan diferentes grosores más allá de la negrita y del texto normal:

- bold: negrita
- normal: elimina la propiedad

**Text-shadow ->** sombreado / sombra alrededor del texto: -4px 3px 5px grey:

- **-4px:** desplazamiento de la sombra eje X
  - + derecha, - izquierda
- **3px:** desplazamiento de la sombra eje Y
  - + izquierda, - derecha
- **5px:** difuminado (cuanto mayor sea el valor, más se expandirá el sombreado)
- **grey:** color

## Tamaño

Algunos elementos en línea, como los enlaces, no se pueden redimensionar, mientras que otros sí, como las imágenes o los botones. Para modificar la forma en la que el navegador considera un elemento, se pueden establecer las siguientes propiedades:

- display: block = ("div")
- display: inline - Si se aplica a varios  seguidos, estos se situarían uno junto a otro.

- display: inline-block - El elemento se establece como un bloque, pero se comporta como un elemento en línea, al estilo de una imagen. Los cambios de tamaño tienen efecto en los elementos con esta propiedad.

# Bordes (se pueden especificar lado en todas)

**Border-width ->** borde 4 lados y, opcionalmente, de forma independiente a cada lado, con las propiedades **border-left-width, border-right-width, border-top-width y border-bottom-width.**

**Border-style ->** estilo de borde:

- **solid** (línea continua)
- **dashed** (línea discontinua)
- **dotted** (con puntos)
- **double** (línea continua doble

**Border-color ->** color del borde

**Border ->** las 3 anteriores juntas: border: 2px solid red

- **2px:** medida
- **solid:** estilo
- **red:** color

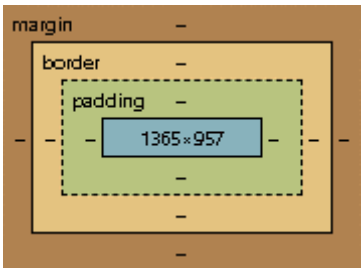
**Border-radius ->** radio de curvatura de las esquinas del borde: border-radius: 5px 8px 5px 8px (la primera se referirá a la esquina superior izquierda y el resto continuarán en el sentido de las agujas del reloj), con 1 solo es para todas

**Border-shadow ->** aplicar sombra a un borde: box-shadow: -5px 5px 8px 0px grey

- **-5px:** eje X
- **5px:** eje Y
- **8px:** radio de difuminado
- **0px:** tamaño que crece la sombra respecto al elemento
- **grey:** color

# Margen

**Margin ->** Representa el espacio entre un elemento y los elementos que lo rodean, o el borde del elemento que lo contiene. (px o %)



**Padding ->** Representa el espacio o margen interno entre un elemento y su contenido. De esta manera, es el elemento contenedor el que fuerza que los elementos interiores se despeguen de su borde. Acepta los mismos valores que la propiedad margin.

## Cosas a tener en cuenta

Ambas propiedades modifican, por defecto, los márgenes exteriores e interiores de los cuatro lados del elemento. Para modificar de manera individual cada uno de los lados, se pueden utilizar los sufijos **-left, -right, -top y -bottom**, o bien establecer las cuatro medidas separadas por espacios, las cuales se aplicarían en el siguiente orden: **arriba, derecha, abajo e izquierda:** margin: 10px 5px 20px 5px

El navegador, por defecto, a la hora de calcular el tamaño que ocupa un elemento, además del tamaño establecido en las propiedades CSS, **como width o height**, también suma el tamaño del margen interno o padding y el grosor del borde al elemento. Por lo tanto, un elemento con un ancho de 200 píxeles, un padding de 20 píxeles y un grosor de borde de 5 píxeles ocupará un ancho total de 250 píxeles (5 píxeles de borde izquierdo, 20 píxeles de padding izquierdo, 200 píxeles de ancho, 20 píxeles de padding derecho y 5 píxeles de borde derecho).

Existe la posibilidad de que las propiedades **width y height** incluyan el margen interno y el grosor del borde, restándolos al espacio disponible para el contenido. En el ejemplo anterior, esto significa que el elemento ocuparía, en este caso, 200 píxeles de ancho, dejando 150 píxeles disponibles para el contenido interno. Para ello, se puede utilizar la propiedad CSS **box-sizing**, aplicando a esta el valor **border-box**, para incluir el borde y el padding al establecer el tamaño que ocupará el elemento.

También existe un valor intermedio para la propiedad box-sizing, que sería **padding-box**. Con esto, incluimos el valor del padding dentro del alto y el ancho que definimos para el elemento, pero no el borde.

Cuando dos elementos contiguos tienen **márgenes verticales** (uno encima del otro), **se separan entre sí con un espacio, que depende del elemento que tenga el mayor margen.** Si ambos tienen un margen de 25 píxeles, se separarán 25 píxeles en total. Sin embargo, cuando se trata de márgenes horizontales (uno junto al otro), **el espacio resultante será la suma de ambos márgenes.** Así, en tal caso, se separarían 50 píxeles en total.

# Posicionamiento (position)

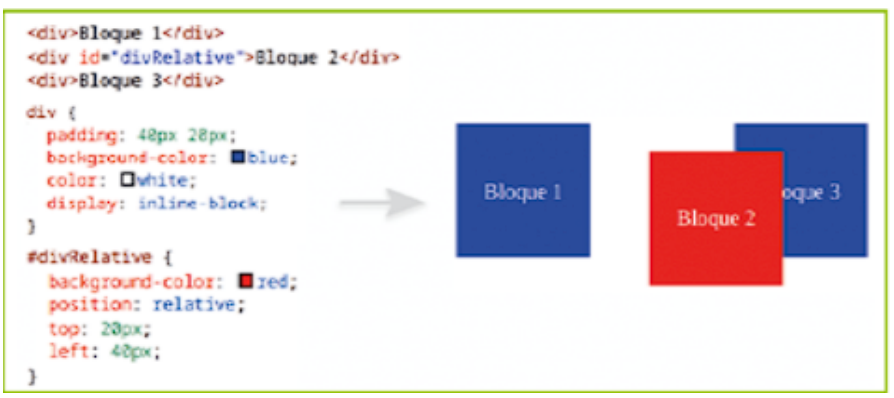
**Static ->** valor por defecto que, después de cada elemento de bloque, realiza un salto de línea para, a continuación, añadir debajo el siguiente elemento, según el orden establecido en la estructura HTML. Los elementos con este posicionamiento no se ven afectados por las propiedades **top, bottom, left y right. Es posible modificar el tamaño ocupado por un elemento añadiéndole márgenes.**

**Relative ->** un elemento con este valor, en su propiedad position, ocupará el mismo espacio que si su valor fuera static.  No obstante, se puede obligar al navegador a desplazar la posición donde lo dibujará **(sin que la posición del resto de elementos se vea alterada), utilizando las propiedades top, bottom, left y right,** que desplazan el elemento una cantidad de píxeles a partir del lado seleccionado. Así, por ejemplo, top: 10px; desplaza el elemento diez píxeles hacia abajo, es decir, lo aleja diez píxeles de su posición con respecto al lado superior (también es posible utilizar valores negativos para conseguir el efecto contrario).

**Absolute ->** Al aplicar este valor, el elemento se sitúa en la esquina superior izquierda del elemento que lo contiene. Además, hay que tener en cuenta que se **«despega»** del documento, es decir, que, para el resto de elementos, la posición se interpreta como si no estuviera ocupando espacio en el documento, de forma similar a si se situara en una capa superior. Esto hará que, en ocasiones, los elementos con posicionamiento absoluto se superpongan a otros elementos.

**Fixed ->** Este valor es muy similar a absolute, con la excepción de que utiliza la ventana, y no el documento, como referencia para posicionarse. **De esta forma, el elemento siempre estará visible (incluso haciendo scroll sobre el documento),** lo que resulta muy útil, por ejemplo, para crear barras de navegación que sean fácilmente accesibles.

**Margin-top / Margin-left ->** Permiten mover un elemento. Sin embargo, si se utilizan las propiedades top, left, right o bottom, dicho elemento deja de situarse con respecto al que lo contiene y emplea todo el documento HTML como referencia. En otras palabras, si se asignan los valores left: 0px; bottom: 20px;, el elemento se situará pegado al lado izquierdo de la página y a veinte píxeles de distancia de la parte inferior del documento.



# Imagen de fondo

```css
#div2 {
  margin: 200px;
  width: 200px;
  background-image: url('../img.png');
  background-position: left top;
  background-repeat: repeat;
  border: 2px solid black;
}
```

**Background-image ->** Su valor será la ruta a la imagen desde el directorio donde se encuentra el archivo CSS. Esta debe incluirse entre **comillas simples y paréntesis, precedida de la palabra url,** así: background-image: url('../img.png');.

**Background-position ->** En el caso de que la imagen sea más pequeña que el elemento, **esta propiedad establece su alineación vertical y horizontal.** Se disponen dos valores separados por espacio: la alineación horizontal, que puede ser **left** (izquierda), **center** (centro) o **right** (derecha), y la vertical, **top** (arriba), **center** (centro) o **bottom** (abajo).

**Background-repeat ->** Si la imagen es más pequeña que el elemento, **se repetirá, por defecto, horizontal y verticalmente** hasta ocupar todo el espacio disponible. Este comportamiento se puede **modificar** con los **valores no-repeat** (no se repite), **repeat-x** (se repite horizontalmente) o **repeat-y** (se repite verticalmente).

# FlexBox

**Flexbox es un módulo de diseño** que se introdujo el 23 de julio de 2009. Es compatible con todos los navegadores web.

**Anteriormente, se usaba la propiedad float** para crear diseños haciendo flotar elementos a la izquierda o a la derecha

**Flexbox permite crear diseños alineando elementos a un solo eje**. El eje puede ser horizontal o vertical. Se utiliza mejor para distribuir el espacio para los elementos en el mismo eje.

1. Crea un contenedor principal **.flex-container,** con la característica **display: flex;**
2. Una vez creado el contenedor primario, los elementos dentro del contenedor primario se convertirán en elementos secundarios, **.flex-item**
3. Los elementos flexibles se organizan en filas de forma predeterminada, pero puede cambiar esto para organizar los elementos en **columnas**.

La **propiedad display** se **utiliza** para **establecer** uno de estos **dos valores**: **flex y inline-flex.**

- Si establece **display en flex**, todos los elementos secundarios dentro del contenedor se convierten en un elemento flexible. Se **organizarán automáticamente en una fila** y **cada elemento tendrá** el **mismo ancho que su contenido**. **El contenedor flexible abarca su propio ancho y cada hijo tiene que caber en el ancho del contenedor.**
- Si estableces **display en inline-flex**, se consigue que el **contenedor flexible** se comporte como un elemento.

**flex-direction** puede cambiar la dirección de horizontal o vertical. Tiene cuatro valores que puedes usar:

1. **row,** es el valor predeterminado en el que flex-direction se establece.
2. **row-reverse**
3. **column,** cambia la dirección verticalmente.
4. c**olumn-reverse**

Por razones de accesibilidad, no debe invertir filas o columnas solo para reordenar el contenido. El HTML no ha cambiado, la dirección visual del contenido ha cambiado.

Para ajustar el tamaño de los elementos flexibles, utilizará los tres valores que aparecen en la propiedad flex

1. **grow:** Esto determinará cómo se **expandirán** los elementos para **llenar el espacio adicional** (si lo hay) en el contenedor flexible.
2. **shrink:** Esto determinará cómo se **encogen** los elementos **cuando no hay suficiente espacio** en el contenedor flexible.
3. **basis:** establecerá el **tamaño inicial** de los elementos flexibles.

Úselo exactamente en ese orden. Esto se usa solo en artículos flexibles. **flex: grow shrink basis**

FlexBox

# Transicion

**Las transiciones son los cambios de un estado a otro.** En este primer tema veremos las transiciones, por lo que sólo vamos de un estado a otro, pero más adelante, veremos que existen animaciones que permiten tener múltiples estados.

```
.element {
  width: 100px;
  height: 100px;
  background: red;
  transition: 2s;
}

.element:hover {
  background: blue;
}
```

Como puedes ver, en este caso el cambio de estilo (la transición) ha sido gradual y más suave. Más adelante explicaremos el código y los detalles técnicos, pero primero aprendamos los conceptos.

**Transicion de entrada y salida ->** Las transiciones de **entrada** se **aplican** cuando el estado inicial cambia al estado final (**se produce la condición: el ratón encima del elemento**), mientras que las transiciones de **salida** se **aplican** cuando se deja de cumplir dicha condición (**se quita el ratón encima del elemento**), volviendo a su estado inicial

Aprovechando los selectores CSS, podemos hacer transiciones de entrada y salida diferentes. Por lo general, la transición CSS se coloca en el elemento (como hicimos en el ejemplo anterior). En ese caso, aplicaría siempre.

- **Transicion de entrada:** hay que tener en cuenta que en su lugar, también es posible añadirla en el bloque **.element:hover**. De esta forma, la transición sólo se aplica cuando el usuario tiene el ratón encima del elemento, por lo que sólo ocurriría en la transición de entrada:

  ```
  .element {
    width: 100px;
    height: 100px;
    background: red;
  }

  .element:hover {
  ```

```
  background: blue;
  transition: 2s;
}
```

La transición del ejemplo anterior se aplica sólo al mover el ratón sobre el elemento (transición de entrada). Si movemos el ratón fuera del enlace, no se produce transición, sino que realiza el cambio de forma brusca.

- **Transicion de salida:** Por otro lado, utilizando **:not(:hover)** podemos invertir la situación y conseguir que sólo se aplique la transición de salida, ya que sólo entrará en efecto cuando el usuario no tenga el ratón encima del elemento:

```
.element {
  width: 100px;
  height: 100px;
  background: red;
}

.element:not(:hover) {
  background: blue;
  transition: 2s;
}
```

También es posible combinar diferentes transiciones utilizando la Cascada de CSS. Si indicamos una transición en el elemento y otra diferente en el :hover, podremos conseguir (por ejemplo) duraciones diferentes:

```
.element {
  width: 100px;
  height: 100px;
  background: red;
  transition: 4s;
}

.element:hover {
  background: blue;
  transition: 1s;
}
```

En este caso, observa que la transición de entrada inicial es muy rápida (1 segundo). Sin embargo, cuando quitamos el ratón, se producirá la transición de salida (4 segundos).

**Transicion por propiedad:** Es posible que queramos indicar (por ejemplo) diferentes duraciones dependiendo de la propiedad CSS. Por ejemplo, que tarde mucho en cambiar el ancho del elemento, pero muy poco en cambiar el color de fondo.

Para ello, podemos separar con comas las diferentes propiedades que queremos transicionar:

```
.element {
  width: 200px;
  height: 200px;
  background: grey;
  transition:
    width 4s,
    background 1s;
}

.element:hover {
  width: 400px;
  background: deeppink;
}
```

Observa que en este caso, **la transición de ancho** (propiedad width de CSS) **tardará 4 segundos en cambiar**, **mientras** que la **transición del color de fondo** (propiedad background de CSS) **tardará** muy poco, **sólo 1s.**

- **Desencadenante de la transición:** En estos ejemplos hemos utilizado :hover como desencadenante de la transición CSS porque es muy sencillo de visualizar en los ejemplos. **Sin embargo, existen muchísimas formas que podemos utilizar para desencadenar una transición:**

**:hover** Cuando mueves el ratón (o dispositivo apuntador) sobre un elemento.

**:active** Cuando estás pulsando con el ratón (o dispositivo apuntador) sobre un elemento.

**:focus** Cuando un elemento gana el foco (por ejemplo, entrar en un campo de texto <input>).

En general, se puede utilizar con cualquier pseudoclase CSS de interacción, pero también podemos utilizarla al añadir clases, por ejemplo, usando un poquito de Javascript:

```
const element = document.querySelector(".element");
element.addEventListener("click", () => element.classList.toggle("light"));
```

```
body {
  background: #141414;
}

.element {
  width: 200px;
  height: 200px;
  background: grey;
  transition: all 2s;
}

.light {
  background: gold;
  box-shadow: 0 0 25px gold;
}
```

En este caso, mediante Javascript estamos utilizando un evento click para añadir o quitar la clase light mediante un .toggle() de clases HTML.

# Transformacion

**Transform:** modificar el espacio de coordenadas de un elemento, lo que permite mover, rotar, escalar o inclinar elementos en dos o tres dimensiones.

1. **Translate:** Mueve un elemento de su posición original.

```
  transform: translate(50px, 100px); /* Mueve el elemento 50px a la derecha y 100px hacia abajo */
```

2. **Rotate:** Rota un elemento en el plano 2D.

```
  transform: rotate(45deg); /* Rota el elemento 45 grados en el sentido de las agujas del reloj */
```

3. **Scale:** Cambia el tamaño de un elemento.

```
transform: scale(1.5); /* Escala el elemento al 150% de su tamaño original */
```

4. **Skew:** Inclina un elemento.

```
transform: skew(30deg, 20deg); /* Inclina el elemento 30 grados en el eje X y 20 grados en el eje Y */
```

5. **Matrix:** Aplica una transformación 2D utilizando una matriz.

```
transform: matrix(1, 0.5, -0.5, 1, 0, 0); /* Aplica una combinación de transformaciones */
```

Puedes combinar múltiples transformaciones en una sola declaración:

```
transform: translate(50px, 100px) rotate(45deg) scale(1.5);
```

# Animacion

# Etiquetas HTML adicionales

`<article>`:

- **Propósito**: Define contenido independiente y autocontenido, como artículos de blog, comentarios, etc.

```
<article>
  <h2>Título del Artículo</h2>
  <p>Contenido del artículo...</p>
</article>
```

`<aside>`:

- **Propósito**: Contenido que está tangencialmente relacionado con el contenido principal, como barras laterales, anuncios, etc.

```
<aside>
  <h2>Información Relacionada</h2>
  <p>Contenido adicional o contextual...</p>
</aside>
```

`<header>`:

- **Propósito**: Representa un contenedor para contenido introductorio o de navegación, como logotipos, menús, encabezados de sección, etc.

```
<header>
  <h1>Encabezado Principal</h1>
  <nav>
    <ul>
      <li><a href="#inicio">Inicio</a></li>
      <li><a href="#sobre-nosotros">Sobre Nosotros</a></li>
    </ul>
  </nav>
</header>
```

`<footer>`:

- **Propósito**: Define un pie de página para su contenido más cercano o para toda la página.

```
<footer>
  <p>&copy; 2024 Mi Empresa</p>
</footer>
```

`<main>`:

- **Propósito**: Representa el contenido principal del documento. Solo debe haber un `<main>` por documento.

```
<main>
  <h1>Contenido Principal</h1>
  <p>Aquí va el contenido principal de la página.</p>
</main>
```

`<figure>` **y** `<figcaption>`:

- **Propósito**: `<figure>` es utilizado para agrupar contenido ilustrativo, como imágenes, gráficos, etc., y `<figcaption>` proporciona una leyenda para el contenido.

```
<figure>

  <img src="imagen.jpg" alt="Descripción de la imagen">
  <figcaption>Descripción de la imagen.</figcaption>

</figure>
```

# Etiquetas CSS adicionales

La pseudo-clase `:hover` en CSS se utiliza para aplicar estilos a un elemento cuando el puntero del ratón está sobre él. Esto permite crear efectos interactivos y mejorar la experiencia del usuario en tu sitio web.

html

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ejemplo de :hover</title>
    <style>
        .boton {
            background-color: blue;
            color: white;
            padding: 10px 20px;
            text-decoration: none;
        }

        .boton:hover {
            background-color: red;
            color: yellow;
        }
    </style>
</head>
<body>
    <a href="#" class="boton">Pasa el ratón sobre mí</a>
</body>
</html>
```

### Explicación

- `.boton`: Define el estilo básico del botón, con un fondo azul y texto blanco.
- `.boton:hover`: Define el estilo cuando el ratón está sobre el botón, cambiando el fondo a rojo y el texto a amarillo.

## Similar a :hover

1. `:active`:
    - **Propósito**: Aplica estilos a un elemento cuando se está haciendo clic sobre él.

```
a:active {
    background-color: green;
}
```

2. `:focus`:
    - **Propósito**: Aplica estilos a un elemento cuando está enfocado, como cuando un usuario hace clic en un campo de formulario o usa la tecla Tab para navegar.

```
input:focus {
    border-color: blue;
}
```

3. `:visited`:
    - **Propósito**: Aplica estilos a enlaces que el usuario ya ha visitado.

```
a:visited {
    color: purple;
}
```

4. `:checked`:
    - **Propósito**: Aplica estilos a elementos de formulario que están seleccionados, como checkboxes y radio buttons.

```
input[type="checkbox"]:checked {
    background-color: yellow;
}
```

5. `:nth-child()`:
    - **Propósito**: Aplica estilos a elementos según su posición en el árbol DOM.

```
ul li:nth-child(odd) {
    background-color: lightgray;
}
```

6. `:before` **y** `:after`:
    - **Propósito**: Permite insertar contenido antes o después del contenido real de un elemento.

```
p:before {
    content: "Nota: ";
    color: red;
}
```

*Estas pseudo-clases y pseudo-elementos permiten crear una variedad de efectos interactivos y estilizar elementos en diferentes estados de interacción, mejorando la experiencia del usuario en la web.*